Guia de Implementação e Localização de Código - GVA-WEB

Propósito: Este documento serve como um guia para o avaliador, mapeando as principais funcionalidades de segurança e de negócio do projeto GVA-WEB aos seus respectivos arquivos e trechos de código na base do projeto.

1. Autenticação e Gestão de Contas

1.1. Registro de Usuário e Confirmação de E-mail

- **Fluxo e Validação:** app/auth/routes.py na função register(). Aqui são validados os dados do formulário, a complexidade da senha, a resposta do CAPTCHA e, após salvar o usuário, é disparada a função de envio do e-mail de confirmação.
- **Formulário e Regras:** app/auth/forms.py na classe RegistrationForm. Contém os campos, os validadores de senha forte (password_complexity) e o aceite dos termos.
- **Lógica de Confirmação:** A rota confirm_email(token) em app/auth/routes.py recebe o token do e-mail, valida-o e ativa a conta do usuário, mudando o campo is_confirmed no modelo.
- **Modelo de Dados:** O campo is_confirmed está na classe User em app/models.py.

1.2. Login e Autenticação de Dois Fatores (2FA)

- **Fluxo e Validação:** app/auth/routes.py na função login(). Esta função verifica a senha e, se o usuário tiver 2FA ativo, redireciona para a rota de verificação.
- **Lógica do 2FA:** A verificação do código de 6 dígitos acontece na rota tfa_verify() em app/auth/routes.py. A ativação do 2FA (com geração de QR Code) ocorre na rota tfa_setup() em app/main/routes.py.
- **Modelo de Dados:** Os campos tfa_secret e tfa_enabled estão na classe User em app/models.py.

1.3. Recuperação e Alteração de Senha

- **Recuperação de Senha:**
    - **Tokens:** A lógica para gerar e verificar os tokens seguros (JWT) está nos métodos get_reset_password_token() e verify_reset_password_token() na classe User em app/models.py.

    - **Rotas:** O fluxo é orquestrado pelas rotas reset_password_request() e reset_password(token) em app/auth/routes.py.
    - **Envio de E-mail:** A função de apoio send_password_reset_email() está em app/auth/email.py.
- **Alteração de Senha (Usuário Logado):**
    - **Rota:** A funcionalidade está implementada na rota change_password() em app/main/routes.py. Ela exige a senha atual para confirmação.
    - **Formulário:** O formulário utilizado é o ChangePasswordForm em app/auth/forms.py.

1.4. Exclusão de Conta pelo Usuário

- **Rota e Lógica:** A funcionalidade está na rota delete_account() em app/main/routes.py, que exige a senha atual do usuário para confirmar a exclusão permanente da conta.
- **Exclusão em Cascata:** A remoção de todos os dados associados ao usuário (locais, alimentos, logs) é garantida pela opção cascade="all, delete-orphan" nos relacionamentos definidos na classe User em app/models.py.

2. Autorização e Painel de Administrador

2.1. Autorização de Acesso

- **Usuário Comum:** A garantia de que um usuário só pode ver/editar/excluir seus próprios dados é implementada nas rotas de CRUD em app/main/routes.py, através de filtros nas consultas ao banco de dados, como Location.query.filter_by(user_id=current_user.id).
- **Administrador:**
    - **Modelo de Dados:** A permissão é definida pelo campo booleano is_admin na classe User em app/models.py.
    - **Proteção de Rotas:** Foi criado um decorador customizado @admin_required no arquivo app/decorators.py. Este decorador é aplicado em todas as rotas do painel de administração em app/admin/routes.py, bloqueando o acesso de usuários não autorizados com um erro 403.
    - **Gerenciamento de Usuários:** As rotas para o admin listar, editar (promover a admin) e excluir usuários estão em app/admin/routes.py.

3. Funcionalidades de Segurança Adicionais

3.1. Sistema de Auditoria

- **Modelo de Dados:** A estrutura dos logs é definida pela classe AuditLog em app/models.py.
- **Lógica de Registro:** Uma função auxiliar log_audit() é chamada em rotas críticas (ex: login(), delete_user(), change_password()) para registrar os eventos no banco de dados.
- **Visualização:** A rota audit_log() em app/admin/routes.py busca todos os registros e os exibe para o administrador.

3.2. Proteção do Lado do Cliente

- **CAPTCHA:** A verificação recaptcha.verify() na rota register() (app/auth/routes.py) e a tag {{ recaptcha }} no template de registro protegem contra bots.
- **CSP e Cabeçalhos de Segurança:** A implementação está centralizada na função add_security_headers() com o decorador @app.after_request, localizada no arquivo principal da aplicação, app/__init__.py.

4. Funcionalidades de Negócio

4.1. Notificações de Validade

- **Comando e Lógica Principal:** A funcionalidade de verificação de produtos próximos ao vencimento é encapsulada como um comando customizado do Flask, send-expiry-alerts, definido no arquivo app/cli.py.
- **Envio de E-mail:** A lógica do comando chama a função de apoio send_expiry_alert_email() (definida em app/auth/email.py) para enviar os alertas.